IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Andrew Harvey Barr et al.

Examiner: Merant, Guerrier

Serial No. 10/685,177

Art Unit: 2117

Filing Date: October 14, 2003

Attorney Docket No.: 200310428-1

Title: CPU Register Diagnostic Testing

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF FILED UNDER 37 C.F.R. § 41.37

Sir:

This appeal brief is being filed with a Notice of Appeal in response to the latest office action dated July 2, 2008.

The Commissioner is hereby authorized to charge requisite fees due for this submission to Deposit Account No. 08-2025 (Hewlett Packard).

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas. The Hewlett-Packard Development Company, L.P., is the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

On information and belief, there are no appeals, interferences, or judicial proceedings known to the appellant, the appellant's legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board of Patent Appeals and Interferences (the "Board") decision in the pending appeal.

III. STATUS OF CLAIMS

A. Total Claims: 1-21

B. Current Status of Claims:

1. Claims canceled: none
2. Claims withdrawn: none
3. Claims pending: 1-21
4. Claims allowed: none
5. Claims rejected: 1-21
6. Claims objected to: none

C. Claims on Appeal: 1-21

As indicated above, claims 1-21 are pending in this application, stand finally rejected, and are being appealed. These claims are rejected in the final office action mailed July 2, 2008 ("the last office action").

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter relates to computer software. More particularly, the claimed subject matter relates to software compilers.

Independent claim 1 recites a method of compiling a software program to be executed on a target central processing unit (CPU) (page 2, lines 3-4). User code instructions are scheduled during said compiling (block 404 in FIG. 4; page 7, lines 19-21; blocks 504 and 506 in FIG. 5; page 8, lines 7-9). Diagnostic testing of CPU registers is opportunistically scheduled during said compiling (block 404 in FIG. 4; page 7, lines 19-21; blocks 504 and 508 in FIG. 5; page 8, lines 7-15) so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling (page 7, lines 21-23). The diagnostic testing (FIG. 7 and page 9, lines 7-22) is such that if an error in the CPU register is indicated, a jump to a fault handler routine occurs (block 712 in FIG. 7 and page 9, lines 20-22).

Independent claim 11 recites a method of compiling a program to be executed on a target central processing unit (CPU) (page 2, lines 3-4). Diagnostic testing of CPU registers is opportunistically scheduled (block 404 in FIG. 4 and page 7, lines 19-21), and a predetermined level of aggressiveness for the scheduling of the register diagnostic testing is checked (block 501 in FIG. 5 and page 8, lines 1-4). A next register to be tested is determined (block 502 in FIG. 5 and page 8, lines 5-7), and a determination is made as to whether there is sufficient opportunity to schedule a register diagnostic instruction to test the next register (block 504 in FIG. 5 and page 8, lines 7-8). If there is sufficient opportunity, then the register diagnostic instruction is scheduled to test the next register (block 508 in FIG. 5 and page 8, lines 9-15). If there is insufficient opportunity, then user code instructions are scheduled (block 506 in FIG. 5 and page 8, lines 8-9).

Independent claim 12 recites a computer-readable medium (fixed storage unit 9 and data storage medium 8 in FIG. 1) embedded with a compiler program for execution on a target microprocessor (CPU 3 in FIG. 1). The computer-readable medium includes executable code which is configured to schedule user code instructions (block 404 in FIG. 4; page 7, lines 19-21; blocks 504 and 506 in FIG. 5; page 8, lines 7-9). The computer-readable medium further includes executable code which is configured to opportunistically schedule diagnostic testing of registers in the microprocessor (block 404 in FIG. 4; page 7, lines 19-21; blocks 504 and 508 in FIG. 5; page 8, lines 7-15) so that testing of the registers occurs within normal running of the user code instructions (page 7, lines 21-23), and such that if an error in the register is indicated, a jump to a fault handler routine occurs (block 712 in FIG. 7 and page 9, lines 7-21).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are to be reviewed on appeal:

1. The rejection of claims 1-3, 6, 12-14 and 21 under 35 U.S.C. §103 (a) as being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138);

2. The rejection of claims 7-10 and 17-20 under 35 U.S.C. §103 (a) as also being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138) and further in view of Raina (US 6,134,675); and

3. The rejection of claims 4, 5, 11, and 15-16 under 35 U.S.C. §103 (a) as also being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138), further in view of Schreiber (US 6,460,173), and further in view of Tirumalai et al. (US 5,835,776).

VII. ARGUMENT

Applicants respectfully traverse the aforementioned rejections of claims 1-21 in the latest office action for the following reasons.

A. Claims 1-3, 6, 12-14 and 21

Claims 1-3, 6, 12-14 and 21 stand rejected under 35 U.S.C. §103 (a) as being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138). This rejection is respectfully traversed.

Claim 1 recites as follows.

> 1. A **method of compiling a software program** to be executed on a target central processing unit (CPU), the method comprising: scheduling user code instructions during said compiling; and **opportunistically scheduling diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling**, and such that if an error in the CPU register is indicated, a jump to a fault handler routine occurs.
>
> (Emphases added.)

As shown above, claim 1 very clearly recites, "A **method of compiling a program** ... comprising: ... **opportunistically scheduling diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling** ...." (Emphases added.) In other words, claim 1 expressly recites that opportunistic scheduling of diagnostic testing of CPU registers is performed during compilation of a software program. This is done "so that testing of the CPU registers occurs within normal running of the user code instructions during execuction of the program after said compiling ...."

Applicants respectfully submit that program compilation is generally applied to source code (for example, written in a programming language such as C, C++, etc) so as to generate an executable file that may be executed on a target microprocessor. Compilation of software programs is discussed in the original application as follows. "FIG. 2 illustrates example software elements of the illustrative computer system of FIG. 1. Shown are application programs **26**. Such applications **26** may be compiled using a compiler **34** incorporated with the teachings of the present invention. The compiled application programs **26** access the runtime libraries **34** for services during execution, which in turn access the operating system **32** for system services. The compiler **34** also accesses the operating system **32** for system services during compilation of application programs **26**." (Page 4, lines 21-28.)

1. Quach in view of Dygon et al. does not teach or suggest the claimed invention as neither reference relates to a "method of compiling a software program"

i. The citation to Quach pertains to instruction issuance during execution, not to a "method of compiling a software program"

The citation to Quach pertains to a different technological art from the claimed invention. The cited "issue module" (col. 2, lines 32-42) of Quach is a hardware unit within a microprocessor. The issue module **issues operation codes** during execution of the executable file. These operation codes may be issued to one or more execution units within the microprocessor.

Hence, the citation to Quach pertains to instruction issuance within a microprocessor, not to a **method of compiling a software program**. In other words, **while Quach's instruction issuance is performed during execution of the executable file by the microprocessor, the claimed "method of compiling" is performed in order to generate an executable file.**

ii. The citation to Dygon et al. pertains to remote testing of software products, not to a "method of compiling a software program"

The citation to Dygon et al. also pertains to a different technological art from the claimed invention. Dygon et al. discloses an **automated test system for testing software products**. In particular, Dygon et al. discloses a system that "allows for the testing process to be conducted remotely, which enables the software products to exist on a variety of computers located at various locations." (Paragraph [0015], emphasis added.)

Hence, the citation to Dygon et al. pertains to remote software testing, not to a **method of compiling a software program**. In other words, **while Dygon et al.'s remote software testing is performed during execution of the executable file, the claimed "method of compiling" is performed in order to generate an executable file.**

In conclusion, as discussed above, neither Quach nor Dygon et al. relate to program compilation. Therefore, applicants respectfully submit that Quach in view of Dygon et al. does not disclose or suggest the claimed "method of compiling a software program."

2. The claimed step of "opportunistic scheduling diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling" is not disclosed or suggested by Quach in view of Dygon et al.

As recited in claim 1, steps performed during program compilation include scheduling user code instructions and **opportunistically scheduling diagnostic testing of CPU registers during said compiling**. These steps are described, for example, on page 7, lines 19-28, which is reproduced below for convenience of reference.

> In accordance with an embodiment of the invention, the scheduling performed **404** includes both the scheduling of user code instructions and the opportunistic scheduling of register diagnostics. In other words, **at compile time, the compiler schedules testing of registers so that the testing occurs within**

> **the normal running of the user code. The testing may be scheduled opportunistically and intelligently by the compiler to reduce the amount of performance overhead incurred. This scheduling is discussed further below in relation to FIG. 5.** This scheduling is performed until a determination **406** is made that the user code is completely scheduled. Thereafter, the scheduling may be considered as completed **408**.
>
> (Page 7, lines 19-28, emphasis added.)

As discussed further below, Quach in view of Dygon et al. does not disclose or suggest the claimed step of "opportunistic scheduling diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling."

> *i. The citation to Quach does not teach or suggest "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling"*

The claimed "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling" is not taught or suggested by the cited disclosure in Quach of a microprocessor operating in a high reliability (HR) execution mode. The latest office action agrees with this conclusion. (Office action, page 4.)

ii. The citation to Dygon et al. does not teach or suggest "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling."

The claimed "**opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling**" is not taught or suggested by the cited disclosure in Dygon et al. relating to the **remote testing of software products**.

The latest office action cites to paragraphs [0018] and [0032] in relation to this claim element. These paragraphs are reproduced below for convenience of reference.

> [0018] The present invention retrieves a test case that has been selected by the user from the Test Definition Database, places it in a queue (herein referred to as the Test Execution Queue), and then executes each test case by remotely running each test case through the product that is the desired object of the particular test case. When the results of each test are returned, they are processed to compare them with the expected or desired output. The final processed results are then stored in the Test Results Database as well as being returned to the user.
>
> ...
>
> [0032] The Remote Execution Engine 212 receives instructions regarding which test cases to execute from the Test Execution Queue 211. The Test Execution Queue 211 stores the test cases that are to be run, the sequence in which they are to be run, and the time for execution. This information is placed in the Test Execution Queue 211 by the user interacting through the web browser user interface. By using Test Execution Queue 211 to store the information about the tests, the user can schedule testing at optimum times. For example, a test case that could take several hours to complete could be placed in the Test Execution Queue 211 and scheduled to execute after business hours so that the results are available when the user arrives for work the following morning.

As seen above, **the above-cited paragraphs do not relate to program compilation, much less disclose or teach the claimed "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling".** In fact, the above citations to Dygon et al. relate to **remote testing** performed **during execution** of an executable file. In contrast, the claimed step relates to **opportunistic scheduling** of instruction **during compilation** of an executable file.

iii. The combination of Quach and Dygon et al. does not reasonably teach or suggest the claimed step of "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling" as neither cited reference relates to instruction scheduling during program compilation.

As discussed above, neither Quach nor Dygon et al. relate to scheduling instructions during program compilation. Nevertheless, the latest office action asserts that combining Quach and Dygon et al. makes the claimed step of "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling" obvious to one of ordinary skill in the art. To support this assertion, the latest office action states as follows.

> … Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to implement the system/method of ***Quach*** in order to *provide an automated test system that has the ability to apply test cases to a large number of products operating in a variety of platforms. Furthermore, it is desirable to have a test system that can automatically collect, store and analyze the results and provide the test engineer with appropriate, meaningful feedback on the performance of the product (e.g. [0014]).*

(Office action, page 4, emphasis in original.)

Applicants respectfully submit that the above explanation in the latest office action fails to explain how combining Quach and Dygon et al. (neither of which relate to program compilation) would make the claimed step of "opportunistic scheduling of diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling" obvious to one of ordinary skill in the art.

In summary, applicants respectfully submit that claim 1 overcomes its rejection for one or more of the reasons discussed above. Claims 2-3 and 6 depend from claim 1 and so also overcome their rejection for one or more of the reasons discussed above.

Claim 12 recites as follows.

> 12. A computer-readable medium embedded with a **compiler program** for execution on a target microprocessor, the computer-readable medium comprising: executable code configured to schedule user code instructions; and **executable code configured to opportunistically schedule diagnostic testing of registers in the microprocessor so that testing of the registers occurs within normal running of the user code instructions**, and such that if an error in the register is indicated, a jump to a fault handler routine occurs.

(Emphasis added.)

Similar to claim 1, claim 12 recites a "**compiler program**" and "**executable code configured to opportunistically schedule diagnostic testing of registers in the microprocessor so that testing of the registers occurs within normal running of the user code instructions**".

Therefore, applicants respectfully submit that claim 12 overcomes its rejection for one or more of the reasons discussed above in relation to claim 1. Claims 13-14 and 21 depend from claim 12 and so also overcome their rejection.

B. Claims 7-10 and 17-20

Claims 7-10 and 17-20 stand rejected under 35 U.S.C. §103 (a) as being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138) and further in view of Raina (US 6,134,675). This rejection is respectfully traversed.

Claims 7-10 also depend from claim 1. Therefore, applicants respectfully submit that claims 7-10 overcome their rejection for one or more of the reasons discussed above in section A.

Claims 17-20 depend from claim 12. Therefore, applicants respectfully submit that claims 17-20 overcome their rejection for one or more of the reasons discussed above in section A.

In addition, each of claims 7 and 17 recites, "the scheduled diagnostic testing includes writing known data to a register, reading data from the register, and comparing the known data with the data that was read." Applicants note that this claim element is performed **by the program compiler**. In contrast, the latest office action cites to Raina in relation to this claim element. Applicants respectfully submit that the disclosure of Raina pertains to **testing a multi-core processor**, not a technique for compiling a software program. Therefore, applicants respectfully submit that claims 7 and 17 overcome its rejections for this further reason. Claims 8-10 and 18-20 depend from claims 7 and 17, respectively, and so also overcome their rejections for this further reason.

C. Claims 4, 5, 11 and 15-16

The rejection of claims 4, 5, 11, and 15-16 under 35 U.S.C. §103 (a) as also being unpatentable over Quach (US 6,640,313) in view of Dygon et al. (US 2003/0140138), further in view of Schreiber (US 6,460,173), and further in view of Tirumalai et al. (US 5,835,776). This rejection is respectfully traversed.

Claims 4 and 5 depend from claim 1. Therefore, applicants respectfully submit that claims 4 and 5 overcome their rejection for one or more of the reasons discussed above in section A.

Independent claim 11 also recites, "A method of compiling a program…." and "opportunistically scheduling diagnostic testing of CPU registers." Therefore, applicants respectfully submit that claim 11 overcomes itsrejection for one or more of the reasons discussed above in section A.

Claims 15 and 16 depend from claim 12. Therefore, similar to claim 12, applicants respectfully submit that claims 15 and 16 also overcome their rejection for one or more of the reasons discussed above in section A.

In addition, claims 4 and 11 each recites, "determining if there is sufficient opportunity to schedule a register diagnostic instruction to test the next register." Claim 15 recites a similar limitation. Applicants note that this claim element is performed **by the program compiler**. In contrast, the latest office action cites to Schreiber in relation to this claim element. Applicants respectfully submit that the disclosure of Schreiber pertains to the **allocation of function units in processor design**, not a technique for compiling a software program. Therefore, applicants respectfully submit that claims 4, 11 and 15 overcome their rejections for this further reason. Claims 5 and 16 depend from claims 4 and 15, respectively, and so also overcome their rejections for this further reason.

VIII. CONCLUSION

For at least the above reasons, applicants respectfully request that the rejections of claims 1-21 be overturned.

Respectfully submitted,
ANDREW HARVEY BARR, ET AL.

Dated: September 2, 2008

/James K. Okamoto, Reg. No. 40,110/
James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O. Box 641330
San Jose, CA 95164
Tel.: (408) 436-2110
Fax.: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:		Dated:	
Express Mail Mailing Number (*optional*):			

CLAIMS APPENDIX

CLAIMS INVOLVED IN THE APPEAL

1. A method of compiling a software program to be executed on a target central processing unit (CPU), the method comprising: scheduling user code instructions during said compiling; and opportunistically scheduling diagnostic testing of CPU registers during said compiling so that testing of the CPU registers occurs within normal running of the user code instructions during execution of the program after said compiling, and such that if an error in the CPU register is indicated, a jump to a fault handler routine occurs.

2. The method of claim 1, further comprising:
checking a predetermined level of aggressiveness for the scheduling of the register diagnostic testing.

3. The method of claim 2, further comprising:
determining a next register to be tested.

4. The method of claim 3, further comprising:
determining if there is sufficient opportunity to schedule a register diagnostic instruction to test the next register without substantially impacting performance of the program.

5. The method of claim 4, further comprising:
if there is sufficient opportunity, then scheduling the register diagnostic instruction to test the next register; and
if there is insufficient opportunity, then scheduling user code instructions.

6. The method of claim 1, wherein the CPU registers comprise registers in a general register file.

7. The method of claim 1, wherein the scheduled diagnostic testing includes writing known data to a register, reading data from the register, and comparing the known data with the data that was read.

8. The method of claim 7, wherein if the comparison indicates a difference, then said jump to the fault handler routine is performed.

9, The method of claim 8, wherein the fault handler routine is configured to halt execution to avoid potential data corruption.

10. The method of claim 8, wherein the fault handler routine is configured to remap activity to the register to an alternate register.

11. A method of compiling a program to be executed on a target central processing unit (CPU), the method comprising:
opportunistically scheduling diagnostic testing of CPU registers;
checking a predetermined level of aggressiveness for the scheduling of the register diagnostic testing;
determining a next register to be tested;
determining if there is sufficient opportunity to schedule a register diagnostic instruction to test the next register;
if there is sufficient opportunity, then scheduling the register diagnostic instruction to test the next register; and
if there is insufficient opportunity, then scheduling user code instructions.

12. A computer-readable medium embedded with a compiler program for execution on a target microprocessor, the computer-readable medium comprising: executable code configured to schedule user code

instructions; and executable code configured to opportunistically schedule diagnostic testing of registers in the microprocessor so that testing of the registers occurs within normal running of the user code instructions, and such that if an error in the register is indicated, a jump to a fault handler routine occurs.

13. The computer-readable program product of claim 12, further comprising executable code configured to check a predetermined level of aggressiveness for the scheduling of the register diagnostic testing.

14. The computer-readable program product of claim 13, further comprising executable code configured to determine a next register to be tested.

15. The computer-readable program product of claim 14, further comprising executable code configured to determine if there is sufficient opportunity to schedule a register diagnostic instruction to test the next register.

16. The computer-readable program product of claim 15, further comprising executable code configured to schedule the register diagnostic instruction to test the next register if there is sufficient opportunity and to schedule user code instructions if there is insufficient opportunity.

17. The computer-readable program product of claim 12, wherein the scheduled diagnostic testing includes writing known data to a register, reading data from the register, and comparing the known data with the data that was read.

18. The computer-readable program product of claim 17, wherein if the comparison indicates a difference, then said jump occurs to said fault handler routine.

19, The computer-readable program product of claim 18, wherein the fault handler routine is configured to halt execution to avoid potential data corruption.

20. The computer-readable program product of claim 18, wherein the fault handler routine is configured to remap activity to the register to an alternate register.

21. The computer-readable program product of claim 13, wherein the aggressiveness level is set at a high level, and wherein the diagnostic testing is scheduled to be performed periodically.

EVIDENCE APPENDIX

There are no documents or items submitted under this section.

RELATED PROCEEDINGS APPENDIX

There are no documents or items submitted under this section.